На правах рукописи

Гаранжа Кирилл Владимирович

Интерактивный синтез реалистичных изображений больших 3D сцен с применением графических процессоров

Специальность 05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание учёной степени

кандидата физико-математических наук

Москва 2014

Работа выполнена в федеральном государственном бюджетном учреждении науки

Институте прикладной математики им М.В. Келдыша Российской академии наук

Научный руководитель: доктор физико-математических наук, профессор Галактионов Владимир Александрович, заведующий отделом №2 компьютерной графики и вычислительной оптики Института прикладной математики им. М.В. Келдыша Российской академии наук.

Официальные оппоненты: доктор технических наук, доцент Турлапов Вадим Евгеньевич, профессор кафедры математического обеспечения ЭВМ факультета вычислительной математики и кибернетики Нижегородского государственного университета им. Н.И.Лобачевского (г. Нижний Новгород)

кандидат физико-математических наук, Игнатенко Алексей Викторович, старший научный сотрудник лаборатории компьютерной графики и мультимедиа факультета вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова (г. Москва)

Ведущая организация: Институт вычислительной математики и Математической геофизики Сибирского отделения РАН

Защита состоится «____» _____ 2014 г. в ____ часов на заседании диссертационного совета Д 002.024.01, созданного на базе ФГБУН Институт прикладной математики имени М.В. Келдыша РАН, расположенного по адресу: 125047 Москва, Миусская пл., д.4.

С диссертацией можно ознакомиться в библиотеке и на сайте ФГБУН Институт прикладной математики им. М.В. Келдыша РАН: www.keldysh.ru

Автореферат разослан «____» _____ 2014 г.

Учёный секретарь
Диссертационного совета,
доктор физико-математических наук Т.А. Полилова

Общая характеристика работы

Актуальность. С момента появления компьютерной графики синтез компьютерных изображений нашёл множество полезных приложений. Наиболее известными являются приложения из сферы киноиндустрии и развлечений, где синтез изображений используется для создания визуально красивых компьютерных игр и спецэффектов кино. Кроме этого приложения компьютерного синтеза изображений можно найти в сфере рекламы, медицине, архитектурном и инженерном дизайне. Во многих этих областях синтезируемое изображение содержит реалистичное глобальное освещение, рассчитанное в виртуальной 3D сцене. Например, в сфере архитектурного дизайна реалистичный синтез изображений используется для ответа на вопрос, как будет выглядеть спроектированное здание или интерьер дома в различных условиях освещения. В сфере рекламы и кинопроизводства компьютерная графика используется в процессе смешивания виртуальных 3D объектов и видеозаписей для визуализации сложных сцен и сценариев, что было бы невозможно или очень дорого с использованием одних лишь съёмок на видеокамеру. Подобные сферы приложения требуют высокого уровня реализма, дающего на выходе изображения сцены с различных ракурсов, визуально неотличимые от реальных фотографий таких же сцен в реальном мире.

Реалистичные алгоритмы визуализации используют математические модели физического переноса света на основе лучевой оптики в моделированной виртуальной сцене для синтеза изображений, которые снимаются с различных ракурсов с учётом свойств камеры наблюдателя. Для большого количества приложений реалистичные алгоритмы визуализации требуют большой вычислительной нагрузки для создания одного изображения. Например, расчёт одного кадра изображения со сложным освещением может занимать несколько часов при использовании одного ядра CPU. Поэтому большинство подобных алгоритмов используются для оффлайн визуализации (т.е. без отклика в режиме реального времени) даже при использовании больших многоузловых вычислительных систем. Отсутствие интерактивности во многих алгоритмах реалистичной визуализации (быстрого отклика в генерации реалистичного изображения) осложняет работу разработчика виртуальной среды.

Также большого количества дополнительных вычислительных ресурсов требует использование динамических сцен, где в любой момент времени любой объект или его часть могут изменить свою форму или положение в мировой системе координат, в которой задаётся виртуальная сцена. Как правило, в алгоритмах визуализации, применяющих реалистичную модель распространения света, используется трассировка лучей, где в свою очередь требуется, чтобы геометрические данные сцены были организованы внутри специальной геометрической

базы данных ГБД (часто называемой ускоряющей структурой УС), которая позволяет ускорять расчёт реалистичного глобального освещения.

Во многих сферах компьютерной графики, включая кинопроизводство, архитектурный и инженерный дизайн, всё чаще используются графические процессоры (GPU) благодаря более быстрым вычислениям (в 10-20 раз) и более высокой пропускной способности внутренней памяти по сравнению с традиционными архитектурами центральных процессоров (CPU). Недостатком GPU является фиксированный размер памяти. В существующих реализациях GPU размер памяти является небольшим (1-12 GB) по сравнению с объёмом оперативной памяти центрального процессора (60-100 GB и более).

Во всех сферах, в которых используется синтез реалистичных изображений, происходит рост требований к объёму детализации моделируемых и визуализируемых виртуальных 3D сцен. Например, современные виртуальные сцены из сфер киноиндустрии и архитектурного проектирования могут занимать от нескольких гигабайт до нескольких сотен гигабайт или нескольких терабайт данных. Для эффективного исполнения все современные реалистичные алгоритмы визуализации требуют хранения всех данных сцены в памяти, наиболее близко располагающейся к процессору. Часто подобное требование физически невозможно осуществить. Физическая память процессора может быть мала для хранения всей сцены целиком, это актуально для графических процессоров и также для центральных процессоров. Подобное ограничение требует разработки сложных структур данных и кэширования для амортизации расходов, связанных с доставкой данных сцены из внешнего хранилища (дискового или сетевого пространства) в оперативную память.

Цель диссертационной работы. Исходя из роста требований задач реалистичной визуализации к сложности сцен, массового распространения вычислительно мощных и ограниченных в объёме памяти графических процессоров возникают следующие цели:

- разработка и программная реализация эффективных алгоритмов построения геометрической базы данных (ускоряющей структуры), обеспечивающей эффективную трассировку лучей в сложных анимированных сценах, в том числе с применением графических процессоров;

- разработка и программная реализация эффективных алгоритмов поиска пересечений лучей с применением графического процессора для массивных сцен, для которых объём данных может превышать размеры физической памяти графического процессора. Алгоритм поиска пересечений должен обеспечивать возможность приложения в любых алгоритмах расчёта глобального освещения, в том числе в интерактивных алгоритмах, и исполняться на массовых графических процессорах.

Научная новизна. Для ускорения процесса построения геометрической базы данных (ГБД) в рамках настоящей работы разработаны следующие ортогональные алгоритмы (гл. 2-5):

1) Алгоритм обновления ГБД в каждом кадре анимации сцены, использующий структуру ранее построенной ГБД (глава 2). Разработанный алгоритм применим для широкого класса анимированных сцен, включая взрывные анимации и структурированное движение.

2) Алгоритм генерации множества кластеров связанных полигонов и использования множества кластеров в качестве строительных блоков в контексте построения ГБД для более быстрого построения, т.к. множество кластеров на порядок меньше множества полигонов (глава 3). Используется предположение, что в процессе анимации связность треугольников сохраняется постоянной, и анимация осуществляется за счёт изменения координат вершин полигонов. Подобное предположение применимо для любых анимированных сцен, в том числе содержащих взрывные анимации, если обломки взрывов и места разрывов обломков рассчитаны заранее.

3) Алгоритмы построения ГБД на графическом процессоре (главы 4 и 5), потребляющие в несколько раз меньше памяти, работающие на порядок быстрее, чем аналоги на CPU и на графическом процессоре. Также решена проблема разрезания полигонов неоднородных размеров в рамках фиксированного заранее определённого бюджета памяти. Подобное разрезание необходимо для построения ГБД, обеспечивающей более эффективный поиск пересечения лучей.

Разработаны масштабируемые алгоритмы поиска пересечений лучей и фильтрации текстур (глава 6), позволяющие с использованием серийных графических процессоров реалистично визуализировать массивные сцены, размер которых может на порядки превышать размер физической памяти графического процессора. Разработанные алгоритмы на порядок быстрее существующих аналогов, исполняющихся на CPU и на графическом процессоре. Разработанные алгоритмы применимы совместно с любыми алгоритмами расчёта глобального освещения, основанными на трассировке лучей, в том числе интерактивных. Разработанный алгоритм поиска пересечений лучей позволил достичь логарифмической зависимости скорости поиска пересечений от размера кэша графического процессора (глава 6). Это позволяет использовать кэш память меньшего размера с сохранением приемлемой скорости поиска пересечений.

Практическая значимость. На основе разработанных алгоритмов с использованием технологи CUDA параллельного программирования на GPU реализован программный комплекс визуализации, применение которого позволяет привести к существенному повышению производительности труда в архитектурном, инженерном проектировании и киноиндустрии.

Разработанный программный комплекс, в частности, позволяет реалистично визуализировать изображения в разрешении Full HD самолёта Боинг 777, состоящего из 360 млн. треугольников в режиме интерактивной визуализации. В качестве метода расчёта глобального освещения использовалась Монте-Карло трассировка путей с несколькими уровнями переотражений, исполняющаяся на серийном графическом процессоре с 3 GB памяти. На расчёт 1 кадра изображения без шума в среднем тратится 3 минуты. Интерактивный режим реалистичной визуализации позволяет осуществлять 10 обновлений изображения в секунду.

Апробация работы. Результаты работы докладывались и обсуждались на:

- Международной конференции “IEEE/EG Ray Tracing Symposium 2008”, США, Лос-Анджелес, 2008;
- Международной конференции “Eurographics Symposium on Rendering 2009”, Испания, Жирона, 2009;
- Международной конференции “Eurographics 2010”, Швеция, Норрчёпинг, 2010;
- Международной конференции “Computer Graphics International 2011”, Канада, Оттава, 2011;
- Международной конференции “High Performance Graphics 2011”, Канада, Ванкувер, 2011;
- Международной конференции “SIGGRAPH 2011”, Канада, Ванкувер, 2011;
- Международной конференции “Graphicon 2011”, Россия, Москва, 2011;
- Семинаре по комп. графике и машинному зрению Ю.М. Баяковского (ф-т ВМК МГУ);
- Семинаре направления «Программирование» им. М. Р. Шура-Бура в ИПМ им. М. В. Келдыша РАН.

Публикации. По результатам работы имеются 7 публикаций, соответствующих требованиям ВАК, из них 3 публикации в научных рецензируемых журналах [2][3][4] входят в библиографическую базу Web of Science и 7 публикаций [1]-[7] входят в библиографическую базу Scopus.

Содержание работы

Во введении рассказывается об актуальности и проблематике расчёта освещения и синтеза реалистичных изображений, в частности, интерактивного синтеза изображений. Приведены основные принципы используемых алгоритмов и спектр существующих аппаратных средств с учётом их достоинств и недостатков. Ставятся цели и задачи диссертационной работы. Описана практическая значимость и научная новизна.

В первой главе диссертации представлен обзор предметной области. Трассировка лучей является основой большинства алгоритмов синтеза реалистичных изображений. В первом разделе описан конвейер визуализации, состоящий из нескольких стадий, в виде которых можно представить алгоритм реалистичной визуализации (см. рис. 1). В отдельной программе интерактивного моделирования (**стадия А**) создаются данные виртуальной сцены, включающие в себя источники света, параметры камеры наблюдателя, геометрическое представление объектов сцены, текстуры, задающие свойства поверхностей объектов. Все эти данные передаются на конвейер визуализации, состоящий из стадий (Б) – (Е).

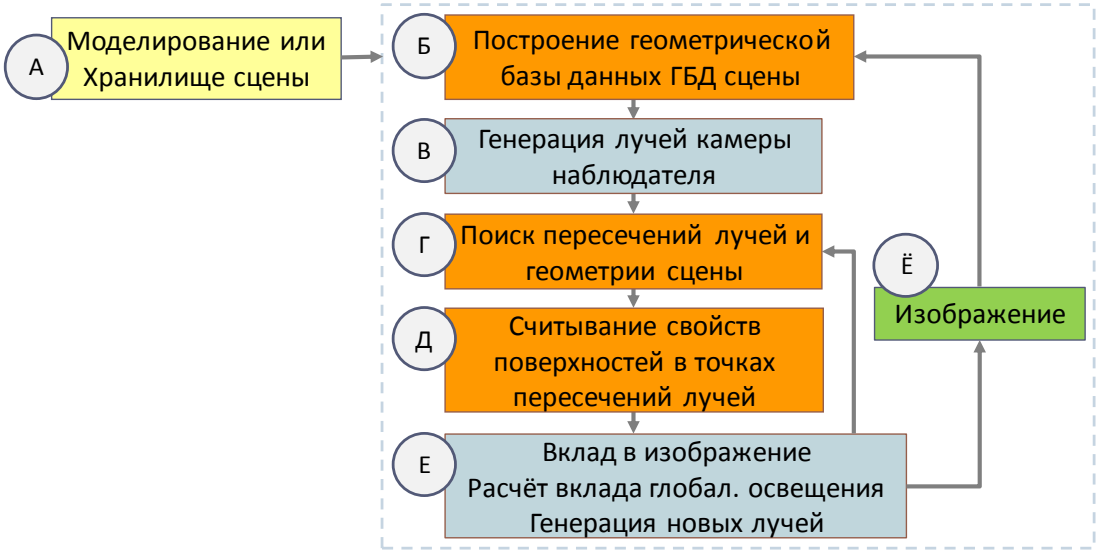

***Рис. 1**: Конвейер реалистичной визуализации. Оранжевым цветом отмечены стадии (Б, Г, Д), на повышение эффективности которых направлен фокус диссертации.*

Во-первых (**стадия Б**), для каждого кадра анимации перед началом расчёта глобального освещения необходимо 1 раз построить геометрическую базу данных (ГБД), которая организует в поисковой структуре геометрическое представление сцены для быстрого осуществления операций пересечения лучей с объектами сцены и поиска ближайших точек пересечения для каждого луча. В дальнейшем в тексте ГБД может называться ускоряющей структурой (УС).

В качестве выходных данных **стадия В** производит множество лучей, берущих начало в положении наблюдателя, с учётом фокусного расстояния и разрешения изображения. В **стадии Г** осуществляется поиск пересечения лучей и всех объектов сцены с использованием ГБД. В **стадии Д** производится считывание свойств поверхностей геометрических объектов в точках пересечения для каждого луча, для которого пересечение найдено. Свойства поверхностей могут задаваться функциями рассеивания и поглощения света, параметры этих функций могут задаваться текстурами. Текстура является 2D дискретным изображением, задающим варьирующиеся параметры. Текстуры отображаются на геометрическую поверхность с помощью текстурных координат, заданных в программе мо-

делирования для каждой контрольной точки поверхности. **Стадия E** осуществляет, с учётом свойств поверхности в точке пересечения луча, расчёты прямого и непрямого освещения, попадающего в данную точку. Для расчёта прямого освещения необходимо проверить блокируются ли лучи света, проведённые от источников света к точке освещения. Если не блокируются (проверяется с помощью поиска пересечений), то производится вклад в часть изображения, соответствующее точке. Непрямое освещение учитывает свет, достигающий этой точки посредством нескольких отражений и преломлений через другие объекты сцены. Рассчитанный вклад глобального освещения для каждого пикселя, записывается в изображение (**Ё**). Существуют алгоритмы, позволяющие визуализировать временные результаты расчётов глобального освещения в каждой итерации с прогрессивным накоплением рассчитанного освещения в каждом пикселе, позволяя таким образом производить интерактивную навигацию в сцене и наблюдать постепенную сходимость к конечному результату.

Во втором разделе представлен алгоритм стохастической трассировки путей, как один из способов решения уравнения визуализации, позволяющий синтезировать реалистичные изображения. В третьем разделе представлен один из способов полигонального представления 3D сцены (самый распространённый), использующийся в настоящей работе. В четвёртом разделе представлены 3 основных вида ускоряющих структур: на основе 3D регулярной сетки (grid), иерархия секущих плоскостей (kd-tree), иерархия охватывающих оболочек (BVH). Примитивы, из которых состоит 3D объект сцены, должны быть организованы в специальной ускоряющей структуре (другое название: геометрическая база данных, ГБД) для того, чтобы во время поиска пересечений лучей можно было ценой осуществления прохода лучей в узлах ускоряющей структуры уменьшить число тестов пересечения лучей и примитивов сцены. Ускоряющая структура является более качественной (или эффективной), если обеспечивает более высокую скорость поиска пересечений лучей. В частности, необходимо, чтобы множество примитивов, соответствующих некоторому узлу ускоряющей структуры, было ограничено компактной охватывающей оболочкой, так чтобы данная оболочка захватывала как можно меньше пустого пространства. Представлены существующие алгоритмы построения ускоряющих структур, в том числе с использованием графического процессора.

В пятом разделе представлен алгоритм поиска пересечения лучей и геометрических объектов сцены с использованием иерархии охватывающих оболочек (BVH), а также обзор ранних работ. В шестом разделе представлена архитектура и ограничения графического процессора. В седьмом разделе представлен обзор существующих алгоритмов визуализации массивных сцен.

Во второй главе описан алгоритм эффективного обновления существующей иерархии охватывающих оболочек (Bounding Volume Hierarchies, сокр. BVH) на основе структуры ранее построенной иерархии BVH (впервые опубликован в статье [1]). Данный алгоритм применяется для организации полигонов внутри дерева BVH различных типов динамических сцен и позволяет производить деревья, обеспечивающие эффективную трассировку лучей. Разработанный алгоритм объединяет достоинства нескольких более ранних методов, адаптивно уменьшая сложность операций перестроения BVH в каждом кадре в процессе анимации сцены. Минимизация сложных операций перестроения BVH в каждом кадре анимации осуществляется благодаря использованию групп треугольников, которые когерентно перемещаются в сцене. Таким группам соответствуют поддеревья BVH, и в процессе обновления структуры BVH переставляется корень всего поддерева с целью минимизации ожидаемого времени трассировки лучей, обеспечиваемого новой структурой BVH. Таким образом, алгоритм эффективно обрабатывает структурированное движение. Также данный алгоритм определяет группы «взорвавшихся» треугольников (т.е. подвергнувшиеся хаотичному движению) для произведения операций перестроения структуры внутри поддерева BVH, соответствующего группе. Эффективная локализация операций перестроения топологии на несколько независимых, непересекающихся поддеревьев BVH в значительной мере уменьшает вычислительную сложность операций перестроения по сравнению с алгоритмом перестроения всей топологии BVH.

В третьей главе представлен алгоритм быстрого построения дерева BVH для динамических 3D сцен с использованием предположения сохранения связности полигонов в сцене (впервые опубликован в статье [2]). Во многих случаях анимация сцены осуществляется при помощи изменения координат вершин полигонов, в то время как связи между треугольниками остаются неизменными (т.е. индексы вершин остаются постоянными в структуре данных каждого треугольника). Подобные анимированные сцены называют деформируемыми. Сцены, содержащие взрывающиеся объекты, можно запрограммировать с сохранением связей треугольников в процессе анимации, если предварительно определить обломки объектов и места разрыва.

Если в геометрической модели, представляющей объект, отношение *Кол-во Вершин / Кол-воТреугольников* < *3*, то среди треугольников модели есть доля связности (т.е. на некоторые вершины ссылаются несколько треугольников). В реальных сценах треугольник не является отдельным объектом – это всего лишь примитив геометрического представления. Даже небольшое число связанных треугольников представляют некоторый объект.

С использованием предлагаемого в этой главе подхода можно ускорить любой алгоритм построения BVH. Ускорение процесса построения BVH достига-

ется при помощи уменьшения количества примитивов, которые учитываются в процессе построения BVH. В качестве примитивов используются кластеры треугольников, ограниченные охватывающими оболочками. В каждом таком кластере содержится несколько связанных треугольников. За счёт этого достигается ускорение стадии построения BVH по сравнению с традиционными алгоритмами построения BVH, где в качестве примитивов используются все треугольники сцены.

Конфигурация кластеров треугольников для каждого динамического объекта сцены предварительно рассчитывается до начала анимации. Полученные кластеры с обновлёнными координатами вершин используются в процессе полного перестроения BVH в каждом кадре анимации сцены. Предполагается, что индексы вершин, на которые ссылаются треугольники, остаются постоянными в процессе анимации. Данное предположение является допустимым для широкого класса анимированных сцен. При этом анимация может быть иерархической, деформируемой или взрывной. Проектирование эвристики, использующейся для генерации кластеров, основано на модели геометрической вероятности и предположении когерентного движения связанных треугольников, принадлежащих одному кластеру.

В четвертой главе описан простой в реализации параллельный алгоритм быстрого построения структуры BVH с использованием графического процессора и технологии CUDA за линейное время (опубликован в статье [4]). Ускорение процесса построения BVH достигается за счёт использования вспомогательной равномерной 3D сетки. Эта сетка строится внутри охватывающей оболочки сцены, а примитивы (в данном случае, треугольники) распределяются по ячейкам сетки один раз во время построения BVH. Благодаря использованию данной сетки производство каждого узла BVH осуществляется с использованием ограниченного сверху числа вычислительных операций и чтения/записи в память.

Также эффективно решается проблема, связанная с «тонкими и длинными» треугольниками, для которых охватывающая оболочка, выровненная по осям координат (сокр. AABB), может повлечь снижение скорости трассировки лучей. Подобные треугольники разрезаются на несколько частей, где каждый частичный треугольник имеет свою охватывающую оболочку, более мелкую, чем базовая оболочка всего треугольника. Причём параллельный метод разделения треугольников на части устроен таким образом, что суммарное количество всех частичных треугольников сцены не превышает заранее выделенного объёма памяти, решая, таким образом, проблему возможного переполнения памяти.

В пятой главе описан более простой в реализации и более мощный вариант алгоритма HLBVH, который назван HLBVH2 (впервые опубликован в статье [6]). Алгоритм Hierarchical Linear Bounding Volume Hierarchies (кратко HLBVH), рабо-

тая на графическом процессоре, демонстрировал возможность строить ускоряющую структуру, необходимую для трассировки лучей, в реальном времени даже для сцен, содержащих миллионы динамических треугольников. В новом алгоритме HLBVH2 сложные структуры хранения префиксных сумм, сжатия и частичного поиска в ширину, необходимые для пространственного распределения, были заменены на конвейер, построенный на основе простой концепции очередей задач и бинарного поиска. Новый алгоритм строит идентичное дерево значительно быстрее (в 5-10 раз), потребляет меньше памяти (в 4 раза) по сравнению с алгоритмом HLBVH.

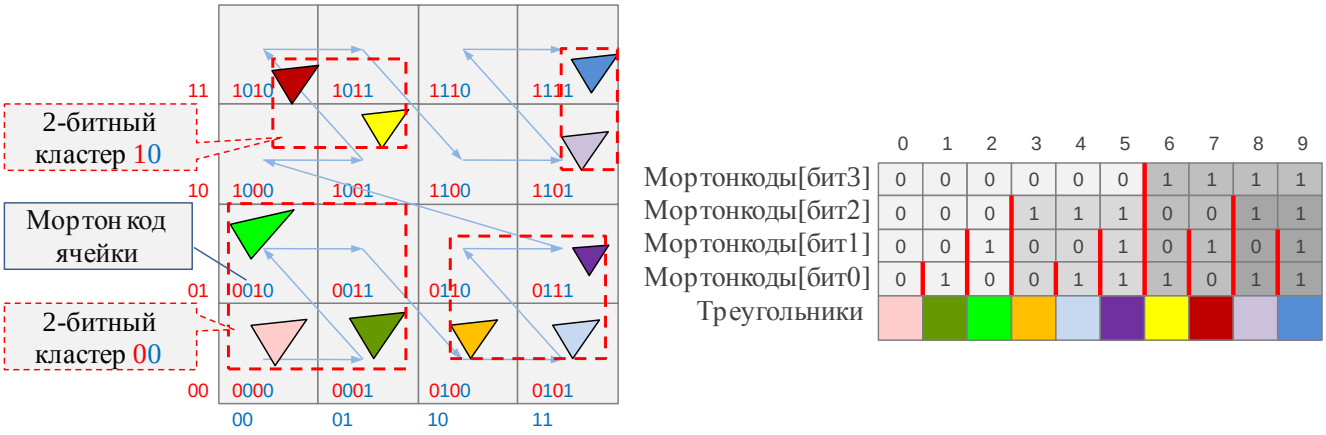

Рис. 2: *Примеры: 2D пространство, дискретизированное с использованием 2 бит (2^2 равных интервалов) в каждом измерении; отсортированный массив треугольников в порядке возрастания их соответствующих 4-битных мортон-кодов.*

Сцена	кол-во треуг-ов	кол-во 15-бит кластер.	Потребление памяти		Время построения	
			во время построения	готовое дерево	HLBVH	HLBVH2
Fairy Forest	174K	2.4k	33Mb	4Mb	23ms	4.8ms
Conference	282K	2.5k	36Mb	6.5Mb	45ms	6.2ms
Stanford Dragon	870K	2.1k	51Mb	20Mb	81ms	8.1ms
Turbine Blade	1760K	2.3k	75Mb	42Mb	137ms	10.5ms
Power Plant	12700K	2.0k	367Mb	290Mb	-	62.1ms

Разбивка по стадиям, время мс	
Сорт. треуг-ов	1.83
Верх. уров.	2.17
Нижн. уров.	3.1
Вычисл AABB	1.0

Таблица 1: *Время построения и потребление памяти алгоритма HLBVH2. В данную таблицу для сравнения также выписано время исполнения раннего алгоритма HLBVH в 2-уровневой версии, обеспечивающего такое же качество BVH, как и 2-уровневая версия HLBVH2. Новый алгоритм позволяет строить BVH на порядок быстрее. Благодаря уменьшенному потреблению памяти в HLBVH2 (в 4 раза по сравнению с HLBVH) стало возможно строить ускоряющую структуру для крупной сцены PowerPlant, состоящей из 12,7 млн. треугольников.*

В основе разработки алгоритма HLBVH2 лежит идея, основанная на сортировке данных по мортон-кодам: область внутри охватывающей оболочки AABB сцены дискретизуется используя n бит (2^n равных интервалов) в каждом из 3 измерений; на треугольнике выбирается одна точка (например, центр), для которой вычисляется мортон-код, содержащий $3n$ бит, который составляется последовательным чередованием бит дискретных координат в x, y, z измерении. Затем тре-

угольники сортируются в порядке возрастания значений их мортон-кодов. После этого строится дерево BVH, организующее треугольники с использованием мортон-кодов и бинарного поиска как метода поиска секущей плоскости, использующегося для разделения узла дерева на два новых узла-потомка.

В шестой главе описан алгоритм трассировки лучей для больших сцен на графическом процессоре. Используются специализированные для конкретной задачи трассировки лучей методы хранения, пересылки и кэширования данных, из которых состоит сцена. Алгоритм поиска пересечений лучей и сцены использует сложную ускоряющую структуру, которая логически организована в виде множества равных страниц данных, которые загружаются в память графического процессора по запросу. В качестве дополнения используется быстрый метод сжатия с потерями геометрических данных с помощью квантования.

Предложенный алгоритм трассировки лучей для больших сцен, реализованный с применением технологии CUDA, применим для различных частных задач, в том числе и для стохастической Монте-Карло трассировки путей, которая порождает лучи с произвольными направлениями, что влечёт за собой необходимость обращаться к произвольным областям памяти в рамках поиска пересечения лучей и сцен для одного пикселя. Подобная ситуация могла бы затруднить эффективное массовое распараллеливание трассировки лучей на графическом процессоре, объём памяти которого строго ограничен. Однако в данной работепродемонстрированы результаты высокой кэш эффективности на графическом процессоре для сцен, объём которых значительно превышает объём памяти графического процессора. Этот алгоритм впервые был представлен на конференции SIGGRAPH 2011 [7], где была продемонстрирована интерактивная трассировка лучей с учётом глобального освещения для массивных сцен, состоящих из нескольких сотен миллионов полигонов, на графическом процессоре ноутбука. Алгоритмы подобного рода в английской литературе имеют приставку out-of-core (т.е. алгоритмы, спроектированные для обработки объёмов данных, не влезающих полностью за один раз в память процессора). Новый алгоритм является out-of-core по отношению к памяти GPU.

Анализ производительности многих программ обработки графики показывает, что программы, исполняющиеся на GPU, могут превзойти в 10-20 раз по скорости исполнения аналогичные программы, исполняющиеся на CPU. Такое ускорение достигается благодаря большей вычислительной мощности современных графических процессоров (более 1 Tflops) и высокой пропускной способности чтения из внутренней памяти GPU (более 150 Гб/с.). Достигнуть подобного ускорения удаётся, если всё рабочее множество данных алгоритма (например, все текстуры и полигональные модели) полностью вмещается в память GPU. Задачи решения уравнения визуализации, использующие трассировку лучей, для каждо-

го пикселя могут порождать запросы в любые области памяти, где хранится сцена. Поэтому возникает необходимость спроектировать алгоритм решения подобных задач для больших сцен в условиях ограниченного размера физической памяти GPU (см. схему работы с большим объёмом данных на рис. 3). Пусть все данные сцены (полигональная модель и текстуры) вмещается в память хоста (память CPU). Размер кэша ограничен размером физической памяти GPU. На GPU выполняется параллельная многопоточная программа, обрабатывающая большое множество данных, запускается в каждой итерации цикла, называемого ***главным циклом обработки данных*** (также см. рис. 3):

```
void out_of_core_data_processing()
{
  int new_data = 1
  while(new_data) {
    // Параллельный многопоточный обработчик данных на GPU
    request_and_process_data_kernel(data)
    // Доставка страниц данных в кэш GPU
    new_data = swap_requested_pages(gpu_data_manager)
  }
}
```

Листинг 1: *Цикл обработки большого объёма данных.*

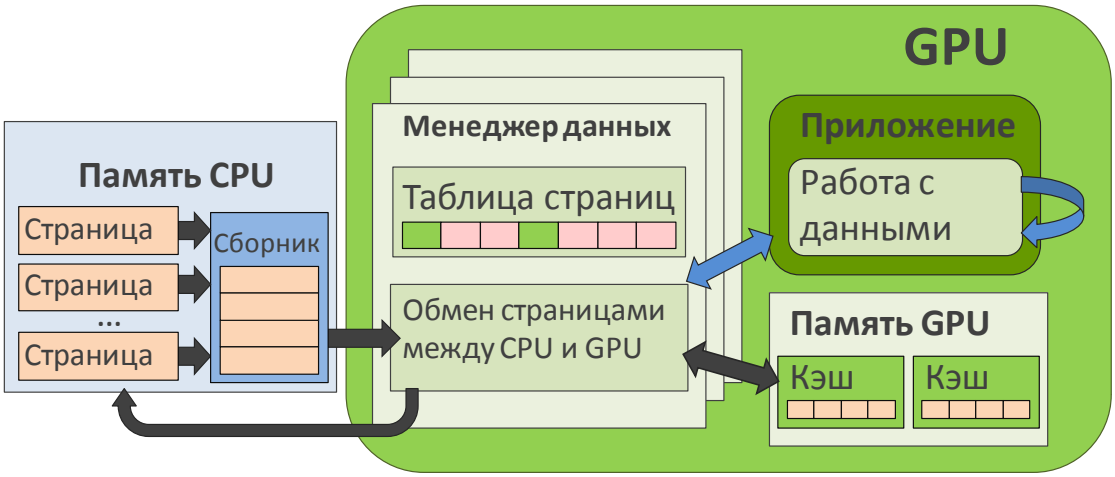

Рис. 3: *Менеджер данных, не влезающих в память графического процессора (GPU), однако влезающих в память центрального процессора (CPU). Все данные организованы в виде набора страниц. Приложение, исполняющееся на GPU, обрабатывает данные в цикле и порождает запросы к одному или нескольким менеджерам данных. Если область данных не находится в кэше GPU, то обработчик данных для конкретного потока откладывается на будущее, а менеджер данных доставляет соответствующие этой области страницы данных в кэш, вытесняя другие страницы. CPU получает массив индексов запрошенных страниц, затем все соответствующие страницы собираются в один непрерывный массив для единоразовой пересылки на GPU одного блока данных, состоящего из многих страниц. Внутри GPU поступившие страницы распределяются в необходимые области кэша и для всех ранее прерванных обработчиков данных возобновляется параллельная обработка данных. Данные о местонахождении страниц записываются в специальной таблице страниц и обновляются во время движения страниц. Цикл обработки данных на GPU прекращает исполнение, когда новые страницы не запрашиваются через менеджер данных.*

Если для некоторого потока в кэше GPU нет страниц, соответствующих запрошенным данным, то вычисления для подзадачи, соответствующей этому потоку, откладываются до следующей итерации цикла (см. Листинг 1). В той же итерации цикла другая параллельная программа, исполняющаяся на GPU swap_requested_pages() производит операции, связанные с политикой пересылки запрошенных страниц данных и обновления соответствующих таблиц страниц. На последующих итерациях цикла запрошенные ранее данные могут находиться в кэше GPU, поэтому возобновление отложенных вычислений для некоторых потоков становится возможным. Процесс запроса, обработки данных и пересылки соответствующих страниц повторяется до тех пор, пока не перестанут появляться новые запросы страниц.

На рис. 4 представлены изображения крупной модели, состоящей из сотен миллионов треугольников, синтезированные с помощью разработанного алгоритма, исполняющегося на единственном графическом процессоре. Одна итерация прогрессивной Монте-Карло трассировки путей с 3 уровнями переотражений рассчитывается за 60-300 миллисекунд (выпускается 1 путь на пиксель, имеется 1024x768 пикселей). Подобная скорость позволяет осуществлять интерактивный предварительный просмотр глобального освещения и навигацию в очень детализированных сценах на стандартном пользовательском компьютере.

Авторы других конкурирующих работ производили попытки визуализировать настолько сложные сцены как Боинг с использованием серверных установок, содержащих десятки узлов. При работе на одном узле такая сложная сцена визуализировалась в течение нескольких десятков часов, а интерактивный предварительный просмотр сложных детализированных ракурсов был невозможен.

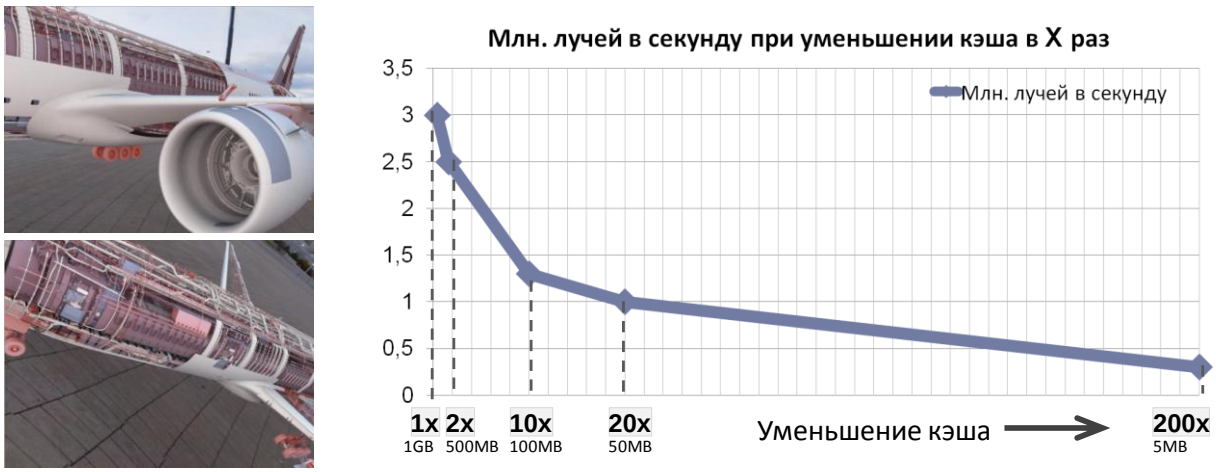

Рис. 4: *Модель самолёта Боинг 777 (предоставлен компанией Боинг, 360 млн. треугольников, занимающих 5Гб в памяти), освещённого картой окружающей среды. На графике представлены замеры скорости трассировки лучей в решении задачи Монте-Карло трассировки путей с 3 уровнями переотражений (стресс-тест для кэша). Замеры времени производились при разных размерах выделенного кэша на GPU: 1 Гб, 500 Мб (кэш в 2 раза меньше максимального), 100 Мб, 50 Мб, 5 Мб. Несмотря на уменьшение размера кэша в 200 раз, скорость трассировки лучей упала всего в 10 раз.*

В заключении представлены примеры практического использования и основные результаты работы, которые состоят в следующем:

1. Разработаны масштабируемые алгоритмы поиска пересечений лучей и фильтрации текстур, позволяющие с использованием серийных графических процессоров реалистично визуализировать массивные сцены (до 1 млрд. треугольников, до 1 терабайта текстур) на порядок быстрее существующих аналогов, исполняющихся на CPU и на графическом процессоре. Разработанные алгоритмы применимы совместно с любыми алгоритмами расчёта глобального освещения, основанными на трассировке лучей, также применимы для интерактивных систем.

2. Разработанный алгоритм поиска пересечений лучей позволил достичь логарифмической зависимости скорости поиска пересечений от размера кэша графического процессора. Это позволяет использовать кэш память меньшего размера с сохранением приемлемой скорости поиска пересечений.

3. Разработано несколько алгоритмов быстрого построения геометрической базы данных, в том числе алгоритмов, исполняющихся на графическом процессоре с более низким и предсказуемым потреблением памяти и на порядок быстрее существующих аналогов.

4. На основе разработанных алгоритмов реализован программный комплекс реалистичной визуализации, применение которого позволяет привести к существенному повышению производительности труда в архитектурном, инженерном проектировании и киноиндустрии.

Рис. 5: *Пример использования разработанных алгоритмов в программном продукте Сентилео реалистичной визуализации. Визуализация финального кадра сцены Боинга 777 (360 млн. треугольников) и New York Times Square (25 млн. треугольников, 10 Гб текстур) осуществляется за 5-10 мин. Также доступен режим интерактивного просмотра глобального освещения со скоростью 5-10 обновлений изображения в секунду.*

Алгоритм поиска пересечений лучей в массивных сценах, представленный в главе 6, а также модифицированные алгоритмы построения геометрической базы данных, представленные в главах 2, 4 и 5, были внедрены в программный продукт Сентилео (см. сайт http://www.centileo.com) в соответствующие стадии конвейера реалистичной визуализации (см. рис. 1). Реализация осуществлена с применением технологии программирования CUDA. Разработанный продукт Сенти-

лео исполняется на массовых настольных и мобильных компьютерах, оснащённых серийными графическими процессорами, например на ноутбуке с графическим процессором GTX 485M, 780M.

На рис.5 представлены примеры синтезированных изображений высокодетализированных сцен, состоящих из текстур высокого разрешения с применением одного графического процессора NVIDIA GeForce GTX 580. Представленные результаты на порядок превосходят по скорости визуализации систему, основанную на платформе трассировки лучей OptiX, исполняющуюся на профессиональном графическом процессоре Quadro 6000, содержащего 6 GB физической памяти. Программный продукт Сентилео по итогам внедрения разработанных в настоящей работе алгоритмов, был представлен на крупнейшей международной выставке индустрии компьютерной графики SIGGRAPH 2012.

Список публикаций

[1] *Garanzha K.* Efficient Clustered BVH Update Algorithm for Highly-Dynamic Models // Proceedings of the IEEE/EG Symposium on Interactive Ray Tracing, pp. 123-130 – 2008 [1]

[2] *Garanzha K.* The Use of Precomputed Triangle Clusters for Accelerated Ray Tracing in Dynamic Scenes // Proceedings of Eurographics Symposium on Rendering 2009 / Computer Graphics Forum, Vol. 28, № 4, pp. 1199-1206 – 2009 [1,2]

[3] *Garanzha K., Loop C.* Fast Ray Sorting and Breadth-First Packet Traversal for GPU Ray Tracing // Proceedings of 31th Annual Conference of the European Association for Computer Graphics Eurographics 2010, Norrköping, Sweden / Computer Graphics Forum, Vol. 29, № 2, pp. 289-298 – 2010 [1,2]

[4] *Garanzha K., Premože S., Bely A., Galaktionov V.* Grid-based SAH BVH construction on a GPU // Proceedings of Computer Graphics International 2011, Ottawa, Canada / The Visual Computer, Vol. 27, № 6-8, pp. 697-706 - 2011 [1,2]

[5] *Garanzha K., Bely A., Galaktionov V.* Simple geometry compression for ray tracing on GPU // Conference proceedings of 21-th International Conference on Computer Graphics and Vision GraphiCon-2011, Moscow State University, pp.107-110 - 2011 [1]

[6] *Garanzha K., Pantaleoni J., McAllister D.* Simpler and Faster HLBVH with Work Queues // Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics HPG'11, Vancouver, BC, Canada, pp.59-64 - 2011 [1]

[7] *Garanzha K., Bely A., Premoze S., Galaktionov V.* Out-of-core GPU ray tracing of complex scenes // In Technical talk Proceedings 38th International conference on computer graphics and interactive techniques ACM SIGGRAPH 2011, Article No.21, Vancouver, Canada - 2011 [1]

[1] Публикация автора, входящая в библиографическую базу Scopus, входящая в перечень ВАК
[2] Публикация автора, входящая в библиографическую базу Web of Science, входящая в перечень ВАК